Reto 4

Diseño e Implementación de un servicio de anuncios utilizando un MOM (Message Oriented Middleware)

23/10/2012

UNIVERSIDAD EAFIT

Luisa Fernanda Querubín Osorio

Jorman Andrés Bustos Gómez

# Descripción

Diseñe e Implemente un sistema de anuncios (advertisement) que permita a un conjunto de clientes obtener mensajes de anuncios de productos o servicios. El sistema tiene dos módulos principales: Proveedor de anuncios (AdFuente), es decir, es desde donde se generan los mensajes y un Cliente de anuncios (AdCliente), el cual recibe de manera sincrónica mensajes enviados a un Canal por un AdFuente.

El sistema debe tener un sistema de gestión que permita:

* Canales y Mensajes: Canales a través de los cuales fluyen los Mensajes originados en un AdFuente (pueden haber varias fuentes en un mismo Canal) hacia uno o más AdCliente
* Envíos: son los mensajes que son enviados por un AdFuente hacia un Canal y que le debe llegar a los AdCliente. Tenga en cuenta la situación cuando los clientes están o no en línea, que supuesto realizar al respecto.

Los Canales son temáticos (deportes, tecnología, noticias, culinaria, bolsa, etc) y deben ser gestionados en el sistema (crear, modificar, borrar, etc un canal).

* Los anuncios son recibidos por los clientes en modo PUSH y PULL. Por anuncios PUSH se entiende la característica de recibir en un cliente mensajes sin haber sido solicitada explícitamente por el cliente. Por anuncios PULL se entiende cuando el cliente explícitamente recupera mensajes de un canal específico.
* Debe tener en cuenta que criterio utilizará para borrar mensajes de una cola.
* Las siguientes son las consideraciones de diseño: Utilice un middleware orientado a mensajes (se recomienda mensajerias en Java según la especificación JMS (Java Message Service) como openjms – http://openjms.sourceforge.net o Apache ActiveMQ http://activemq.apache.org o puede utilizar otra mensajería como la de Microsoft o IBM o muchas otras.

Defina los siguientes aspectos:

* Análisis, Modelo y Arquitectura de la aplicación
* Entidades, Componentes/Paquetes
* Niveles y capas
* Diagramas UML necesarios

# Requisitos funcionales y no funcionales del sistema distribuido

Requisitos Funcionales:

* Elementos en una red de datos.
* La comunicación de estos deben ser por el paso de mensajes.

Requisitos No funcionales:

* Concurrencia: Varias personas  (usuarios) pueden acceder al mismo tiempo a la aplicación de anuncios sin afectar el rendimiento de este. El servicio que este va a ofrecer un servicio multiusuario puesto que debe atender varios AdClientes.
* Transparencia: Los usuarios no pueden ver la lógica del negocio ya que esta se aísla del cliente y el cliente solo interactúa con la interfaz de los anuncios de los canales.
* Tolerancia a fallas: el sistema es tolerante a fallos ya que si se le cae un proceso a un cliente puede entrar en estado de retrasmisión para volver a un estado ‘disponible’ para aceptar los mensajes.
* Escalable: El sistema es escalable ya que se le puede ir aumentando clientes, funcionalidades, etc. y esto no afectara su rendimiento.
* Seguro: Al mandar un mensaje entre publicador y el servidor este no sea inferido o alterado.

# Requisitos de la Aplicación

Requisitos Funcionales:

* Crear Canal
* Modificar Canal
* Borrar Canal
* Obtener Mensaje- Pull
* Enviar Mensaje – Push
* Publicar Mensaje
* Suscribirse a un canal

Requisitos no funcionales:

* Fiabilidad: La aplicación es tolerante a fallos ya que si un cliente no recibe publicaciones entonces entra en un estado de retransmisión para que vuelva a recibir mensajes de manera asincrónica.
* Usabilidad: La aplicación es fácil de entender para el usuario pues la interfaz sería muy sencilla y básicamente es solo publicar un anuncio o simplemente recibir anuncios.
* Eficiencia: La aplicación sería en tiempo real y entonces sería un servicio rápido; no tardaría más de 1 segundos en actualizar los anuncios de cada canal (suponiendo que estamos en un entorno determinista).
* Portabilidad: La aplicación se puede correr e instalar en cualquier lugar, ya que no es un dispositivo físico como tal, simplemente hay que crear la red y acceder a ella y a un canal específico para recibir los anuncios.
* Usuario: La aplicación la pueden usar personas que deseen obtener información acerca de productos y servicios de un tema de algún canal existente.
* Entorno: La aplicación puede ser utilizada en cualquier lugar e incluso desde dispositivos móviles para averiguar por el estado o por anuncios de interés.

# Modelo de interacciones

* Modelo: P2P
* Arquitectura: Cliente/ Servidor
* Capas: Consiste en 2 capas, una capa de presentación que está alojada en la parte del ‘AdCliente’ para ver las publicaciones por canales y la capa lógica del negocio que se comporta como un proveedor que va a estar regulando el paso de publicaciones de la cola y enviándolas a los diferentes clientes que estén conectados a un canal y publicando anuncios que recibe para así enviarlas al cliente.

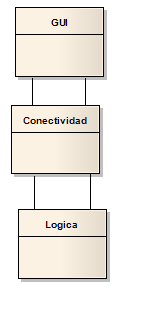
# Características de las interacciones

* Sincrónico: El Proveedor de anuncios (AdFuente) genera los mensajes y los envía a un Cliente (AdCliente), que recibe de manera sincrónica mensajes que un Proveedor ha enviado por un Canal.
* Manejo de estado: El anunciador maneja estado puesto que el cliente puede recibir o no los mensajes los estados son ‘pull y ‘push’.
* Manejo de sesión: Se maneja una sesión puesto que el cliente puede estar conectado o no conectado.

# Definición de middleware adecuado al diseño

Para la implementación de la Aplicación, se usa un Middleware Orientado a Mensajes, en este caso JMS (Java Message Service).

# Niveles y capas



# Diagramas UML necesarios

**AdFuente:** Es el Proveedor que se encarga de generar los mensajes y ponerlos en un Canal para que los Clientes puedan leerlos.

* **AdFuenteInterface:** Es la interface en la que el Proveedor de Mensajes puede crear y eliminar Tópicos, Mensajes y Canales
* **Main:** Entidad principal que permita que el programa se pueda ejecutar
* **Sender:** Permite el envío de los Mensajes por medio de los Canales para que el Cliente los pueda observar.
* **SenderInterface**: Encargada de establecer la conexión entre el Cliente y el Proveedor

**AdCliente:** Es el Cliente que puede suscribirse a un Proveedor recibir los Mensajes que éste envíe por los canales.

* **ClienteInterface:** Es la interface en la que el Proveedor de Mensajes puede suscribirse y visualizar los Mensajes.
* **ConnectInterface:** Encargada de establecer la conexión entre el Cliente y el Proveedor.
* **Receiver:** Permite la recepción de los Mensajes que ha publicado el Proveedor.
* **Main:** Entidad principal que permita que el programa se pueda ejecutar.

